Instituto Tecnológico y de Estudios Superiores de Monterrey

Programación de estructuras de datos y algoritmos fundamentales (TC1031.2)

Profesor: Daniel Pérez Rojas

Act 5.2 - Actividad Integral sobre el uso de códigos hash

(Evidencia Competencia)

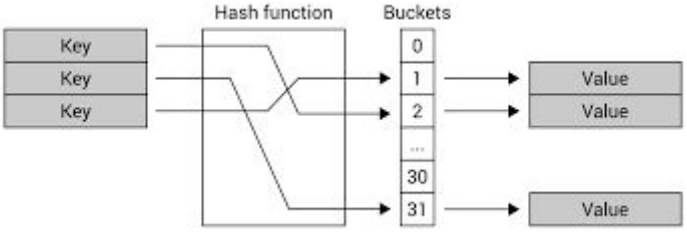
Daniela Hernández y Hernández A01730397

29 de noviembre de 2020

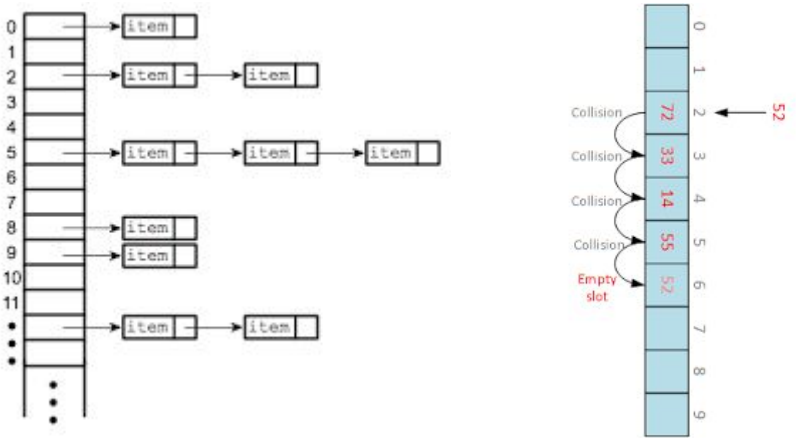
Puebla, Pue.

IMPORTANCIA Y EFICIENCIA DEL USO DE TABLAS HASH

Las tablas hash son una estructura de datos que almacena información de manera asociativa. En este tipo de estructuras, los datos se almacenan en arreglo, donde cada valor de datos tiene su propia "llave" o valor de índice único, generada a través de una función predefinida. La ventaja de este tipo de ordenamiento es que el acceso o búsqueda de los datos se vuelve muy rápido si conocemos el índice de los datos deseados. Lo mismo pasa con operaciones como inserción y eliminación, las cuales son muy rápidas independientemente del tamaño del input, teniendo por tal razón una complejidad de tiempo promedio constante O(1).



La técnica utilizada para obtener el índice único de cada dato es conocida como hash Usualmente, se realiza alguna operación aritmética con la llave y luego se utiliza el operador de módulo para obtener un rango de valores válidos para el tamaño del arreglo. Debido a la naturaleza del hashing, es posible que más de un dato coincida con el mismo valor del índice. En este caso, existen varios acercamientos para manejar las colisiones, entre los más comunes está 'chaining', que consiste en ocupar listas ligadas para insertar valores con índices repetidos, o 'linear probing', que consiste en encontrar el siguiente espacio vacío en el arreglo. Sin embargo, lo que se busca es obtener una función de hash que evite lo más posible las colisiones, ya que de lo contrario, se necesitaría hacer una búsqueda lineal para realizar las operaciones, lo cual impactaría negativamente en la complejidad de tiempo. Sin embargo, una tabla hash sigue siendo mucho más eficiente que otros métodos o estructuras.



Las tablas hash son ampliamente utilizadas en las ciencias computacionales e ingeniería de software debido a su simplicidad de implementación y escalabilidad. Algunas de las aplicaciones de las tablas hash en ámbitos de la vida real son los resúmenes de mensaje

(encripción), verificación de contraseñas, operaciones del compilador, el algoritmo de Rabin-Karp o la vinculación de archivos y rutas juntos (Thakral, 2018). Dentro de estas aplicaciones surge otro concepto relevante, el SHA-256. Este es un algoritmo criptográfico desarrollado por la Agencia de Seguridad Nacional de los Estados Unidos (NSA) y el National Institute of Standards and Technology (NIST), y surge como versión mejorada del SHA-1. SHA-256 pertenece a una familia de algoritmos conocidos como SHA-2, los cuales buscan generar hashes o códigos únicos con base en un estándar que asegure documentos o datos informáticos frente a cualquier agente externo que quiera modificarlos, garantizando la privacidad e integridad del contenido en el procesamiento de información. Un algoritmo de este tipo funciona en una sola dirección, es decir que de cualquier contenido podemos generar su hash, pero de un hash no hay forma de generar el contenido asociado a él, por lo que es considerado 'irreversible' (Dominguez, 2015).

El algoritmo SHA-256 es popular gracias a su equilibrio entre seguridad y coste computacional de generación, pues es un algoritmo muy eficiente y cuenta con una alta resistencia de colisión. Otra de sus particularidades es que la longitud del hash resultante es siempre igual, no importa lo extenso que sea el contenido que uses para generar el hash. Esto sucede porque la manera en que se construye, puesto que comienza con un cifrado de bloque que toma como entrada un conjunto de bits de ancho fijo y genera otro bloque de bits del mismo tamaño. Posteriormente, se utiliza un esquema de relleno criptográficamente seguro, y se aplica iterativamente el cifrado de bloque a una combinación de algún número de bits, más un valor de inicialización o la salida del cifrado de bloque anterior. Debido a que el cifrado de bloque funciona en un conjunto de bits de tamaño fijo, la complejidad de tiempo de ejecución es constante, es decir, O(1). Asimismo, considerando hay un total de $\Theta(n)$ aplicaciones de ese cifrado de bloque (la entrada se divide en bloques de tamaño fijo, por lo que hay $\Theta(n)$ de esos bloques), y el costo de calcular el bit de relleno es igualmente O(1), lo que significa que el tiempo de ejecución de calcular estas funciones hash es $\Theta(n)$, lo que tiene sentido porque cada bit se visita al menos una vez y el trabajo realizado por bit es constante.

Los cifrados de bloque generalmente se implementan de una manera que hace que tomen exactamente la misma cantidad de tiempo para ejecutarse en cualquier combinación de bits de entrada, o al menos, algo muy cercano a la misma cantidad de tiempo, para tratar de hacerlos resistentes a la sincronización. Como resultado, sería muy inusual si estas funciones hash tomaran diferentes cantidades de tiempo para completarse en diferentes entradas del mismo tamaño. Entonces, independientemente del hecho de que el tiempo de ejecución sea $\Theta(n)$, no debería haber mucha variación en el tiempo de ejecución.

Por otra parte, entre las desventajas de este algoritmo se encuentran que su desempeño puede variar entre cada función y puede producir resultados erráticos dependiendo de la longitud de la llave a utilizar. No obstante, es muy ocupado en entornos de blockchain y criptomonedas, ya que se sigue considerando uno de los métodos más seguros ante ataques cibernéticos. Esto nos demuestra que es un algoritmo muy robusto y utilizado en sistemas de alto nivel, por lo que ocuparlo para situaciones pequeñas no sería adecuado, sino que gastaría más recursos de lo necesario.

Referencias:

- Tutorials Point. (2020). *Data Structure and Algorithms - Hash Table*. Recuperado de: https://www.tutorialspoint.com/data_structures_algorithms/hash_data_structure.htm
- Thakral, K. (2018). *Applications of Hashing.* Recuperado de: https://www.geeksforgeeks.org/applications-of-hashing/
- Dominguez, J. (2015). *¿Qué es SHA-256?*. Recuperado de: https://academy.bit2me.com/sha256-algoritmo-bitcoin/
- Rachmawati, D. (2018). *A comparative study of Message Digest 5(MD5) and SHA256 algorithm.* Recuperado de: https://iopscience.iop.org/article/10.1088/1742-6596/978/1/012116/pdf